CAPÍTULO 15
CLASSE ABSTRATA

Uma classe abstrata é uma classe que não é concebida para criar instâncias. Seu propósito é servir como uma superclasse para outras classes.

Depende da regra de negócio transformar ou não a superclasse em uma classe abstrata, no exemplo da empresa que possui funcionários de vários tipos, gerentes, vendedores, secretarias, etc. Não há um nenhum objeto na empresa que seja somente um funcionário e que também não seja uma instância de uma subclasse mais específica. Esse tipo de classe, que não são concebidos para criar objetos, mas servem somente como superclasses, são conhecidas como classes abstratas.

As classes que não são abstratas são chamadas de classes concretas.

As classes são declaradas abstratas inserindo a palavra-chave `abstract` no cabeçalho da classe.

```
public abstract class Funcionario {

    //atributos e métodos omitidos

}
```

15.1. MÉTODOS ABSTRATOS

Até agora, a utilização da superclasse `Funcionario` ajudou a evitar a duplicação de código nas classes Gerente e Vendedor e tornou mais fácil adicionar novos tipos de funcionários no futuro. A classe `Funcionario` se tornou abstrata e, portanto não é possível criar instâncias.

Vamos considerar que cada funcionário do nosso sistema tem uma regra totalmente diferente para ser bonificado, faz algum sentido ter o método `calculaBonificacao` na classe `Funcionario`? Toda subclasse de `Funcionario` terá que reescrever o método de acordo as suas regras e parece inútil a permanência deste método na subclasse.

O problema de excluir é que não poderíamos chamar o método apenas com uma referência a um `Funcionario`, pois ninguém garante que essa referência aponta para um objeto que possui esse método.

Um método abstrato é caracterizado por dois detalhes:

- Ele é prefixado com a palavra-chave `abstract`.
- Ele não tem um corpo de método. Em vez disso, seu cabeçalho termina com um ponto-e-vírgula.

```
public abstract double calculaBonificacao() ;
```

15.2. PROPÓSITOS DA CLASSE ABSTRATA

Declarar uma classe abstrata serve a vários propósitos:

- Nenhuma instância pode ser criada das classes abstratas. Tentar utilizar a palavra-chave `new` com uma classe abstrata é um erro.

- Somente classes abstratas podem ter métodos abstratos. Isso assegura que todos os métodos nas classes concretas sempre possam ser executados. Se permitíssemos um método abstrato em uma classe concreta, poderíamos criar uma instância de uma classe em que faltaria uma implementação para um método.

- As classes abstratas com métodos abstratos forçam as subclasses a sobrescreverem e implementarem esses métodos declarados como abstratos. Se a subclasse não implementar o método ocorre um erro.

```
public class Gerente extends Funcionario {
```

E uma das opções do Eclipse é adicionar a implementação dos métodos abstratos da superclasse.

```
public class Gerente extends Funcionario {
    private d
    Add unimplemented methods
    Make type 'Gerente' abstract
```

A classe abstrata define um modelo para uma funcionalidade e fornece uma implementação incompleta (a parte genérica dessa funcionalidade) que é compartilhada por um grupo de classes derivadas (subclasses). Cada uma das classes derivadas completa a funcionalidade da classe abstrata adicionando um comportamento específico.

Agora, podemos ver o propósito dos métodos abstratos: embora não forneçam uma implementação, não obstante eles asseguram que todas as subclasses tenham um implementação desse método. Em outras palavras, mesmo que a classe `Funcionario` não implemente o método `calculaBonificacao`, ela assegura que todos os funcionários existentes tenham um método `calculaBonificacao` implementado. Isso é feito assegurando que:

- nenhuma instância da classe `Funcionario` possa ser criada diretamente,; e
- que todas as subclasses concretas implementem o método `calculaBonificacao`.

Embora não possamos criar uma instância de uma classe abstrata diretamente, podemos de outro modo utilizar normalmente uma classe abstrata como um tipo. Por

exemplo, as regras normais de polimorfismo permitem tratar gerentes e vendedores como instâncias da classe `Funcionario`. Assim, as partes do sistema que não precisam saber se elas estão lidando com uma subclasse específica podem, em vez disso, utilizar a superclasse.

15.3. EXERCÍCIO: Classe e Métodos abstratos

1) No mesmo projeto da aula anterior. Para isso copie o seu projeto *ds1-jsf-heranca* para a área de trabalho.

2) Importe o projeto ds1-jsf-heranca:
   a. Acesse o menu **File ➔ Import**
   b. Selecione **General ➔ Existing Project into Workspace** e **Next**
   c. Clique em **Browse** e selecione a pasta do projeto
   d. Clique em **Ok** e, depois, em **Finish**.

3) Altere a classe `Funcionario` que se encontra no pacote *br.com.etec.aulaheranca.modelo* para se tornar uma classe abstrata. Basta adicionar a palavra-chave `abstract`.

```
public abstract class Funcionario
```

4) Altere o método `calculaBonificacao` da classe `Funcionario` para abstrato. Basta adicionar a palavra-chave `abstract` e retirar sua implementação.

```
public abstract double calculaBonificacao() ;
```

5) Altere o método `calculaBonificacao` da classe `Gerente` e implemente com a nova regra da empresa. Os gerentes recebem 40% do salário como bonificação.

```
public double calculaBonificacao() {
    return super.getSalario() * 0.4;
}
```

6) Altere o método `calculaBonificacao` da classe `Vendedor` e implemente com a nova regra da empresa. Os vendedores recebem 20% do salário como bonificação.

7) Salve o arquivo e execute novamente a página através da URL: http://localhost:8080/ds1-jsf-heranca/bonificacao.xhtml . Verifique que o polimorfismo foi possível porque o método `calculaBonificacao`, mesmo abstrato continua na classe `Funcionario` e garante que toda subclasse terá que implementar garantindo o tipo.

Referência Bibliográfica:

Apostila Caelum - http://www.caelum.com.br/
http://www.globalcode.com.br/home
http://www.k19.com.br/downloads/apostilas-java
http://www.algaworks.com/
Barnes, David J. e Kolling, Michael. Programação Orientada a Objetos com Java. Pearson
Arnold, Ken, Gosling, James e Holmes, David. A Linguagem de Programação Java. Bookman
Harvey M. Deitel, Paul J. Deitel. Java Como Programar. 6ed. Pearson
Flanagan, David. Guia Essencial do Java. Bookman
Martin, Fowler. UML Essencial.
Livro de Informática – Centro Paula Souza